A Review of Fast ℓ_1-Minimization Algorithms for Robust Face Recognition *

Allen Y. Yang, Arvind Ganesh, Zihan Zhou, S. Shankar Sastry, and Yi Ma †

Abstract

ℓ_1-minimization refers to finding the minimum ℓ_1-norm solution to an underdetermined linear system $\boldsymbol{b} = A\boldsymbol{x}$. It has recently received much attention, mainly motivated by the new compressive sensing theory that shows under quite general conditions the minimum ℓ_1-norm solution is also the sparsest solution to the system of linear equations. Although the underlying problem is a linear program, conventional algorithms such as interior-point methods suffer from poor scalability for large-scale real world problems. A number of accelerated algorithms have been recently proposed that take advantage of the special structure of the ℓ_1-minimization problem. In this paper, we provide a comprehensive review of five representative approaches, namely, *Gradient Projection*, *Homotopy*, *Iterative Shrinkage-Thresholding*, *Proximal Gradient*, and *Augmented Lagrange Multiplier*. The work is intended to fill in a gap in the existing literature to systematically benchmark the performance of these algorithms using a consistent experimental setting. In particular, the paper will focus on a recently proposed face recognition algorithm, where a sparse representation framework has been used to recover human identities from facial images that may be affected by illumination, occlusion, and facial disguise. MATLAB implementations of the algorithms described in this paper have been made publicly available.

*This work was partially supported by NSF IIS 08-49292, NSF ECCS 07-01676, ONR N00014-09-1-0230, ARO MURI W911NF-06-1-0076, and ARL MAST-CTA W911NF-08-2-0004. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute for Government purposes notwithstanding any copyright notation hereon. An extended abstract of the work was previously published in the Proceedings of the International Conference on Image Processing in 2010 [51].

†A. Yang and S. Sastry are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA. A. Ganesh, Z. Zhou, and Y. Ma are with the Coordinated Science Laboratory, University of Illinois, Urbana, USA. Y. Ma is also with the Visual Computing Group, Microsoft Research Asia, Beijing, China. Corresponding author: Allen Yang, Cory Hall, University of California, Berkeley, CA 94720. Email: yang@eecs.berkeley.edu. Tel: 1-510-643-5798. Fax: 1-510-643-2356.

1 Introduction

Compressive sensing (CS) has been one of the hot topics in the signal processing and optimization communities in the last five years or so. In CS theory [11, 19, 10, 13], it has been shown that the minimum ℓ_1-norm solution to an underdetermined system of linear equations is also the sparsest possible solution under quite general conditions. More specifically, suppose there exists an unknown signal $\boldsymbol{x}_0 \in \mathbb{R}^n$, a measurement vector $\boldsymbol{b} \in \mathbb{R}^d$ ($d < n$), and a measurement matrix $A \in \mathbb{R}^{d \times n}$ such that A is full rank and $\boldsymbol{b} = A\boldsymbol{x}_0$. Recovering $\boldsymbol{x}_0$ given A and $\boldsymbol{b}$ constitutes a non-trivial linear inversion problem, since the number of measurements in $\boldsymbol{b}$ is smaller than the number of unknowns in $\boldsymbol{x}_0$. A conventional solution to this problem is the *linear least squares*, which finds the minimum ℓ_2-norm solution (or the solution of least energy) to this system. However, if $\boldsymbol{x}_0$ is sufficiently sparse[1] and the sensing matrix A is *incoherent* with the basis under which $\boldsymbol{x}_0$ is sparse (i.e., the identity matrix in the standard form), then $\boldsymbol{x}_0$ can be exactly recovered by computing the minimum ℓ_1-norm solution, as given by the following optimization problem:

$$(P_1): \quad \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subj. to } \boldsymbol{b} = A\boldsymbol{x}. \tag{1}$$

We refer to the above problem as ℓ_1-minimization or ℓ_1-min. The sparsity-seeking property of (P_1) has been shown to have applications in geophysics, data compression, image processing, sensor networks, and more recently, in computer vision. The interested reader is referred to [11, 2, 10, 48, 52] for a comprehensive review of these applications.

The (P_1) problem can be recast as a linear program (LP) and can be solved by conventional methods such as interior-point methods. However, the computational complexity of these general-purpose algorithms is often too high for many real-world, large-scale applications. Alternatively, heuristic greedy algorithms have been developed to approximate (P_1). *Orthogonal matching pursuit* (OMP) [17] and *least angle regression* (LARS) [22] are two examples in this category. Empirically, these greedy algorithms work better when $\boldsymbol{x}_0$ is very sparse, but will deviate from the solution of (P_1) when the number of non-zero entries in $\boldsymbol{x}_0$ increases, as illustrated in [42]. In other words, the greedy methods do not come with strong theoretical guarantees for global convergence.

Besides scalability, another important requirement for real-world applications is robustness to noise, namely, the observation vector $\boldsymbol{b}$ may be corrupted by data noise. To take into account of the noise, one can relax the equality constraint as follows:

$$(P_{1,2}): \quad \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subj. to } \|\boldsymbol{b} - A\boldsymbol{x}\|_2 \leq \epsilon, \tag{2}$$

where $\epsilon > 0$ is a pre-determined noise level. If the observation vector $\boldsymbol{b}$ is assumed to be corrupted by white noise of magnitude up to ϵ, then the ground-truth signal $\boldsymbol{x}_0$ can be well approximated by $(P_{1,2})$, dubbed *basis pursuit denoising* (BPDN) in [13, 12].

[1] By sparse, we mean that most entries in the vector are zero.

Based on the context of the data noise, the ℓ_2-norm used in the penalty term can also be replaced by other ℓ_p-norms. In this paper, we also focus on a special case of (P_1) given by:

$$(P_{1,1}): \quad \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subj. to } \|\boldsymbol{b} - A\boldsymbol{x}\|_1 \leq \epsilon. \tag{3}$$

This formulation has been used in [49, 47] for robust face recognition. Although $(P_{1,1})$ can be solved by any algorithm designed for (P_1), as we will later explain, its special structure can be further exploited to develop more scalable methods.

In light of the high interest in finding more efficient algorithms to solve these problems, many new algorithms have been proposed. Although it is impossible to summarize all existing algorithms in the literature, in this paper, we provide a comprehensive review of five representative methods, namely, *Gradient Projection* (GP) [23, 31], *Homotopy* [41, 34, 21], *Iterative Shrinkage-Thresholding* (IST) [16, 14, 26, 50], *Proximal Gradient* (PG) [38, 39, 5, 6], and *Augmented Lagrange Multiplier* (ALM) [8, 53]. Although the paper is mainly focused on fast implementations of ℓ_1-min, the reader may refer to [10, 45] for a broader discussion about recovering sparse solutions via other approaches, such as greedy pursuit-type algorithms.

This paper intends to fill in a gap in the existing literature to systematically benchmark the performance of these algorithms using a fair and consistent experimental setting. Due to the attention given to compressive sensing and ℓ_1-minimization, other more sophisticated solutions continue to be developed at a rapid pace. More recent developments include *subspace pursuit* [15], CoSaMP [37], *approximate message-passing* algorithm [20], and *Bregman iterative algorithm* [54], to name a few. However, we do not believe there exists an overall winner that could achieve the best performance in terms of both speed and accuracy for all applications. Therefore, in addition to extensive simulations using synthetic data, our experiments will be focused on a specific application of robust face recognition proposed in [49], where a sparse representation framework has recently been developed to recognize human identities from facial images. To aid peer evaluation, all algorithms discussed in this paper have been made available on our website as a MATLAB toolbox:

`http://www.eecs.berkeley.edu/~yang/software/l1benchmark/`

1.1 Notation

For a vector $\boldsymbol{x} \in \mathbb{R}^n$, we denote by $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ the vectors that collect the positive and negative coefficients of $\boldsymbol{x}$, respectively:

$$\boldsymbol{x} = \boldsymbol{x}_+ - \boldsymbol{x}_-, \boldsymbol{x}_+ \geq 0, \boldsymbol{x}_- \geq 0. \tag{4}$$

We also denote

$$X = \text{diag}(x_1, x_2, \cdots, x_n) \in \mathbb{R}^{n \times n} \tag{5}$$

as a square matrix with the coefficients of $\boldsymbol{x}$ as its diagonal and zero otherwise. The concatenation of two (column) vectors will be written following the

MATLAB convention: $[\boldsymbol{x}_1;\boldsymbol{x}_2] \doteq [\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}]$; $[\boldsymbol{x}_1,\boldsymbol{x}_2] \doteq [\begin{smallmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2\end{smallmatrix}]$. We denote by $\mathbf{1}$ a vector whose components are all one with dimension defined within the context. We represent the Euclidean or ℓ_2-norm by $\|\cdot\|_2$ and the ℓ_1-norm by $\|\cdot\|_1$. The notation $\|\cdot\|$ represents the ℓ_2-norm for vectors and the spectral norm for matrices.

For any real-valued differentiable function $f(\cdot)$, we denote its gradient by $\nabla f(\cdot)$. If a real-valued convex function $g(\cdot)$ is not differentiable everywhere, we represent its subgradient by $\partial g(\cdot)$, defined as follows:

$$\partial g(\boldsymbol{x}) = \{\eta \in \mathbb{R}^n : g(\bar{\boldsymbol{x}}) - g(\boldsymbol{x}) \geq \eta^T(\bar{\boldsymbol{x}} - \boldsymbol{x}), \forall \bar{\boldsymbol{x}} \in \mathbb{R}^n\}. \tag{6}$$

1.2 Primal-Dual Interior-Point Methods

We first discuss a classical solution to the ℓ_1-min problem (P_1), called the *primal-dual interior-point* method, which is usually attributed to the works of [24, 29, 35, 36, 32]. For the sake of simplicity, we assume here that the sparse solution $\boldsymbol{x}$ is nonnegative.[2] Under this assumption, it is easy to see that (P_1) can be converted to the standard primal and dual forms in linear programming:

$$\begin{array}{rll} & \textbf{Primal (P)} & \qquad\qquad & \textbf{Dual (D)} \\ \min_{\boldsymbol{x}} & \boldsymbol{c}^T\boldsymbol{x} & \max_{\boldsymbol{y},\boldsymbol{z}} & \boldsymbol{b}^T\boldsymbol{y} \\ \text{subj. to} & A\boldsymbol{x} = \boldsymbol{b} & \text{subj. to} & A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c} \\ & \boldsymbol{x} \geq 0 & & \boldsymbol{z} \geq 0, \end{array} \tag{7}$$

where for ℓ_1-min, $\boldsymbol{c} = \mathbf{1}$. The primal-dual algorithm simultaneously solves for the primal and dual optimal solutions [9].

It was proposed in [24] that (P) can be converted to a family of logarithmic barrier problems[3]:

$$(P_\mu): \qquad \begin{array}{rl} \min_{\boldsymbol{x}} & \boldsymbol{c}^T\boldsymbol{x} - \mu\sum_{i=1}^n \log x_i. \\ \text{subj. to} & A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} > 0 \end{array} \tag{8}$$

Clearly, a feasible solution $\boldsymbol{x}$ to (P_μ) cannot have zero coefficients. Therefore, we define the interiors of the solution domains for (P) and (D) as:

$$\begin{array}{rcl} P_{++} & = & \{\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} > 0\}, \\ D_{++} & = & \{(\boldsymbol{y},\boldsymbol{z}) : A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \boldsymbol{z} > 0\}, \\ S_{++} & = & P_{++} \times D_{++}. \end{array} \tag{9}$$

Assuming that the above sets are non-empty, it can be shown that (P_μ) has a unique global optimal solution $\boldsymbol{x}(\mu)$ for all $\mu > 0$. As $\mu \to 0$, $\boldsymbol{x}(\mu)$ and $(\boldsymbol{y}(\mu), \boldsymbol{z}(\mu))$ converge to optimal solutions of problems (P) and (D) respectively [35, 36].

[2] This constraint can be easily removed by considering the linear system $\boldsymbol{b} = [A, -A][\boldsymbol{x}_+;\boldsymbol{x}_-]$, where $[\boldsymbol{x}_+;\boldsymbol{x}_-]$ is also nonnegative.

[3] In general, any smooth function Ψ that satisfies $\Psi(0^+) = -\infty$ is a valid barrier function [27].

The primal-dual interior-point algorithm seeks the domain of the *central trajectory* for the problems (P) and (D) in S_{++}, where the central trajectory is defined as the set $S = \{(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu)) : \mu > 0\}$ of solutions to the following system of equations:

$$\begin{array}{c} XZ\mathbf{1} = \mu\mathbf{1},\ A\boldsymbol{x} = \boldsymbol{b}, A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \\ \boldsymbol{x} \geq 0,\ \boldsymbol{z} \geq 0. \end{array} \tag{10}$$

The above condition is also known as the Karush-Kuhn-Tucker (KKT) conditions for the convex program (P_μ) [36, 32].

Hence, the update rule on the current value $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)})$ is defined by the Newton direction $(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{z})$, which is computed as the solution to the following set of linear equations:

$$\begin{array}{rcl} Z^{(k)}\Delta\boldsymbol{x} + X^{(k)}\Delta\boldsymbol{z} & = & \hat{\mu}\mathbf{1} - X^{(k)}\boldsymbol{z}^{(k)}, \\ A\Delta\boldsymbol{x} & = & 0, \\ A^T\Delta\boldsymbol{y} + \Delta\boldsymbol{z} & = & 0, \end{array} \tag{11}$$

where $\hat{\mu}$ is a penalty parameter that is generally different from μ in (P_μ).

In addition to the update rule (11), an algorithm also needs to specify a stopping criterion when the solution is close to the optimum. For ℓ_1-min, some simple rules can be easily evaluated:

1. The relative change of the sparse support set becomes small;
2. The relative change (in the sense of the ℓ_2-norm) of the update of the estimate becomes small;
3. The relative change of the objective function becomes small.

A more detailed discussion about choosing good stopping criteria in different applications is postponed to Section 3.

Algorithm 1 summarizes a conceptual implementation of the interior-point methods.[4] For more details about how to choose the initial values $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{z}^{(0)})$ and the penalty parameter $\hat{\mu}$, the reader is referred to [32, 36]. Algorithm 1 requires a total of $O(\sqrt{n})$ iterations, and each iteration can be executed in $O(n^3)$ operations for solving the linear system (11).

In one simulation shown in Figure 1, the computational complexity of Algorithm 1 with respect to (w.r.t.) the sensing dimension d grows much faster than the other five algorithms in comparison. For example, at $d = 1900$, the fastest algorithm in this simulation, i.e., DALM, only takes about 0.08 sec to complete one trial, which is more than 250 times faster than PDIPA. For this reason, basic BP algorithms should be used with caution in solving real-world applications. The details of the simulation are explained later in Section 3.

[4] There are multiple versions of the primal-dual interior-point solver implemented in MATLAB. Notable examples include SparseLab at `http://sparselab.stanford.edu/`, the CVX environment at `http://cvxr.com/cvx/`, and the $\ell_1$magic package at `http://www.acm.caltech.edu/l1magic/`.

Algorithm 1 Primal-Dual Interior-Point Algorithm (PDIPA)

Input: A full rank matrix $A \in \mathbb{R}^{d\times n}$, $d < n$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, initialization $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{z}^{(0)})$. Iteration $k \leftarrow 0$. Initial penalty μ and a decreasing factor $0 < \delta < \sqrt{n}$.

1: **repeat**
2: $k \leftarrow k + 1$, $\mu \leftarrow \mu(1 - \delta/\sqrt{n})$.
3: Solve (11) for $(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{z})$.
4: $\boldsymbol{x}^{(k)} \leftarrow \boldsymbol{x}^{(k-1)} + \Delta\boldsymbol{x}, \boldsymbol{y}^{(k)} \leftarrow \boldsymbol{y}^{(k-1)} + \Delta\boldsymbol{y}, \boldsymbol{z}^{(k)} \leftarrow \boldsymbol{z}^{(k-1)} + \Delta\boldsymbol{z}$.
5: **until** stopping criterion is satisfied.

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

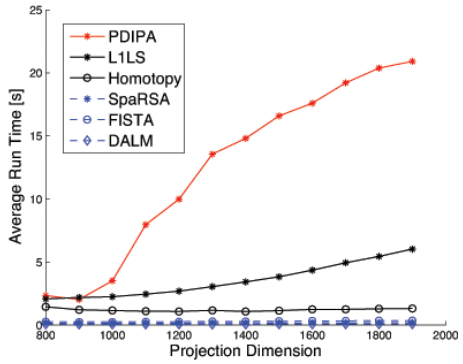


Fig. 1: Average run time of PDIPA in comparison with five other fast algorithms. The simulation setup: $n = 2000$, $k = 200$. The projection matrices are randomly generated based on the standard normal distribution with the dimension varies from 300 to 1900. The support of the ground truth $\boldsymbol{x}_0$ is randomly selected at each trial, and the nonzero coefficients are sampled from the normal distribution.

Next, we will review the five fast ℓ_1-min algorithms shown in Figure 1, namely, *Gradient Projection* in Section 2.1, *Homotopy* in Section 2.2, *Iterative Shrinkage-Thresholding* in Section 2.3, *Proximal Gradient* in Section 2.4, and *Augmented Lagrange Multiplier* in Section 2.5. The algorithms of L1LS, Homotopy, and SpaRSA are provided by their respective authors. We have also provided our implementation of FISTA and DALM on our website.

2 Fast ℓ_1-Min Algorithms

2.1 Gradient Projection Methods

We first discuss *Gradient Projection* (GP) methods that seek a sparse representation $\boldsymbol{x}$ along a certain gradient direction, which induces much faster convergence speed. The approach reformulates the ℓ_1-min problem as a quadratic

programming (QP) problem.

We start with the ℓ_1-min problem $(P_{1,2})$. It is equivalent to the so-called LASSO problem [44]:

$$(LASSO): \quad \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 \quad \text{subj. to } \|\boldsymbol{x}\|_1 \leq \sigma, \tag{12}$$

where $\sigma > 0$ is an appropriately chosen constant. Using a Lagrangian formulation, the problem $(LASSO)$ (and hence, $(P_{1,2})$) can be rewritten as an unconstrained optimization problem:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{13}$$

where λ is the Lagrangian multiplier.

In the literature, there exist two slightly different methods that formulate (13) as a quadratic programming problem, namely, *gradient projection sparse representation* (GPSR) [23] and *truncated Newton interior-point method* (TNIPM) [31].[5]

To formulate the GPSR algorithm, we separate the positive coefficients $\boldsymbol{x}_+$ and the negative coefficients $\boldsymbol{x}_-$ in $\boldsymbol{x}$, and rewrite (13) as

$$\begin{array}{l} \min_{\boldsymbol{x}} Q(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - [A, -A][\boldsymbol{x}_+; \boldsymbol{x}_-]\|_2^2 + \lambda \mathbf{1}^T(\boldsymbol{x}_+ + \boldsymbol{x}_-) \\ \quad \text{subj. to } \boldsymbol{x}_+ \geq 0, \boldsymbol{x}_- \geq 0. \end{array} \tag{14}$$

Problem (14) can be rewritten in the standard QP form as

$$\begin{array}{rl} \min & Q(\boldsymbol{z}) \doteq \boldsymbol{c}^T\boldsymbol{z} + \frac{1}{2}\boldsymbol{z}^T B\boldsymbol{z} \\ \text{subj. to} & \boldsymbol{z} \geq 0, \end{array} \tag{15}$$

where $\boldsymbol{z} = [\boldsymbol{x}_+; \boldsymbol{x}_-]$, $\boldsymbol{c} = \lambda\mathbf{1} + [-A^T\boldsymbol{b}; A^T\boldsymbol{b}]$, and

$$B = \begin{bmatrix} A^TA & -A^TA \\ -A^TA & A^TA \end{bmatrix}. \tag{16}$$

We note that the gradient of $Q(\boldsymbol{z})$ is defined as

$$\nabla_{\boldsymbol{z}} Q(\boldsymbol{z}) = \boldsymbol{c} + B\boldsymbol{z}. \tag{17}$$

This leads to a steepest-descent algorithm that searches from each iterate $\boldsymbol{z}^{(k)}$ along the negative gradient $-\nabla Q(\boldsymbol{z})$:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} - \alpha^{(k)}\nabla Q(\boldsymbol{z}^{(k)}), \tag{18}$$

where $\alpha^{(k)}$ is the step size. This can be solved by a standard *line-search* process [28]. For example, in [23], the direction vector $\boldsymbol{g}^{(k)}$ is defined as

$$g_i^{(k)} = \begin{cases} (\nabla Q(\boldsymbol{z}^{(k)}))_i, & \text{if } z_i^{(k)} > 0 \text{ or } (\nabla Q(\boldsymbol{z}^{(k)}))_i < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

[5] A MATLAB implementation of GPSR is available at `http://www.lx.it.pt/~mtf/GPSR`. A MATLAB Toolbox for TNIPM called L1LS is available at `http://www.stanford.edu/~boyd/l1_ls/`.

Then the step size for the update is chosen to be

$$\alpha^{(k)} = \arg\min_{\alpha} Q(\boldsymbol{z}^{(k)} - \alpha \boldsymbol{g}^{(k)}), \tag{20}$$

which has a closed-form solution

$$\alpha^{(k)} = \frac{(\boldsymbol{g}^{(k)})^T \boldsymbol{g}^{(k)}}{(\boldsymbol{g}^{(k)})^T B \boldsymbol{g}^{(k)}}. \tag{21}$$

The computational complexity and convergence of GPSR is difficult to estimate exactly [23]. Another issue is that the formulation of (15) doubles the dimension of the equations from (13). Therefore, the matrix operations involving B must take into account its special structure w.r.t. A and A^T.

The second GP algorithm, which we will benchmark in Section 3, is *truncated Newton interior-point method* (TNIPM) [31]. It transforms the same objective function (13) to a quadratic program but with inequality constraints:

$$\begin{array}{rl} \min & \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \sum_{i=1}^n u_i. \\ \text{subj. to} & -u_i \leq x_i \leq u_i, \quad i = 1, \cdots, n \end{array} \tag{22}$$

Then a *logarithmic barrier* function for the constraints $-u_i \leq x_i \leq u_i$ is constructed as follows [24]:

$$\Phi(\boldsymbol{x}, \boldsymbol{u}) = -\sum_i \log(u_i + x_i) - \sum_i \log(u_i - x_i). \tag{23}$$

Over the domain of $(\boldsymbol{x}, \boldsymbol{u})$, the central path consists of the unique minimizer $(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))$ of the convex function

$$F_t(\boldsymbol{x}, \boldsymbol{u}) = t(\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \sum_{i=1}^n u_i) + \Phi(\boldsymbol{x}, \boldsymbol{u}), \tag{24}$$

where the parameter $t \in [0, \infty)$.

Using the primal barrier method discussed in Section 1.2, the optimal search direction using Newton's method is computed by

$$\nabla^2 F_t(\boldsymbol{x}, \boldsymbol{u}) \cdot \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{u} \end{bmatrix} = -\nabla F_t(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^{2n}. \tag{25}$$

Again, for large-scale problems, directly solving (25) is computationally expensive. In [31], the search step is accelerated by a *preconditioned conjugate gradients* (PCG) algorithm, where an efficient preconditioner is proposed to approximate the Hessian of $\frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$. The reader is referred to [30, 40] for more details about PCG.

2.2 Homotopy Methods

One of the drawbacks of the PDIPA method is that they require the solution sequence $\boldsymbol{x}(\mu)$ to be close to a "central path" as $\mu \to 0$, which sometimes is

difficult to satisfy and computationally expensive in practice. In this section, we review an approach called *Homotopy methods* [41, 34, 21] that can mitigate these issues.

We recall that $(P_{1,2})$ can be written as an unconstrained convex optimization problem:

$$\begin{aligned} \boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) &= \arg\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \\ &\doteq \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}) \end{aligned} \tag{26}$$

where $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2$, $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$, and $\lambda > 0$ is the Lagrange multiplier. On one hand, w.r.t. a fixed λ, the optimal solution is achieved when $\mathbf{0} \in \partial F(\boldsymbol{x})$. On the other hand, similar to the interior-point algorithm, if we define

$$\mathcal{X} \doteq \{\boldsymbol{x}_\lambda^* : \lambda \in [0, \infty)\}, \tag{27}$$

$\mathcal{X}$ identifies a solution path that follows the change in λ: when $\lambda \to \infty$, $\boldsymbol{x}_\lambda^* = 0$; when $\lambda \to 0$, $\boldsymbol{x}_\lambda^*$ converges to the solution of (P_1).

The Homotopy methods exploit the fact that the objective function $F(\boldsymbol{x})$ undergoes a homotopy from the ℓ_2 constraint to the ℓ_1 objective in (26) as λ decreases. One can further show that the solution path $\mathcal{X}$ is piece-wise constant as a function of λ [41, 22, 21]. Therefore, in constructing a decreasing sequence of λ, it is only necessary to identify those "breakpoints" that lead to changes of the support set of $\boldsymbol{x}_\lambda^*$, namely, either a new nonzero coefficient added or a previous nonzero coefficient removed.

The algorithm operates in an iterative fashion with an initial value $\boldsymbol{x}^{(0)} = 0$. In each iteration, given a nonzero λ, we solve for $\boldsymbol{x}$ satisfying $\partial F(\boldsymbol{x}) = 0$. The first summand f in (26) is differentiable: $\nabla f = A^T(A\boldsymbol{x} - \boldsymbol{b}) \doteq -\boldsymbol{c}(\boldsymbol{x})$. The subgradient of $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ is given by:

$$\boldsymbol{u}(\boldsymbol{x}) \doteq \partial\|\boldsymbol{x}\|_1 = \left\{ \boldsymbol{u} \in \mathbb{R}^n : \begin{matrix} u_i = \operatorname{sgn}(x_i), x_i \neq 0 \\ u_i \in [-1, 1], x_i = 0 \end{matrix} \right\}. \tag{28}$$

Thus, the solution to $\partial F(\boldsymbol{x}) = 0$ is also the solution to the following equation:

$$\boldsymbol{c}(\boldsymbol{x}) = A^T\boldsymbol{b} - A^TA\boldsymbol{x} = \lambda\boldsymbol{u}(\boldsymbol{x}). \tag{29}$$

By the definition (28), the sparse support set at each iteration is given by

$$\mathcal{I} \doteq \{i : |\boldsymbol{c}_i^{(l)}| = \lambda\}. \tag{30}$$

The algorithm then computes the update for $\boldsymbol{x}^{(k)}$ in terms of the direction and the magnitude separately. Specifically, the update direction on the sparse support $\boldsymbol{d}^{(k)}(\mathcal{I})$ is the solution to the following system:

$$A_{\mathcal{I}}^T A_{\mathcal{I}} \boldsymbol{d}^{(k)}(\mathcal{I}) = \operatorname{sgn}(\boldsymbol{c}^{(k)}(\mathcal{I})), \tag{31}$$

where $A_{\mathcal{I}}$ is a submatrix of A that collects the column vectors of A w.r.t. $\mathcal{I}$, and $\boldsymbol{c}^{(k)}(\mathcal{I})$ is a vector that contains the coefficients of $\boldsymbol{c}^{(k)}$ w.r.t. $\mathcal{I}$. For the

coefficients whose indices are not in $\mathcal{I}$, their update directions are manually set to zero. Along the direction indicated by $\boldsymbol{d}^{(k)}$, there are two scenarios when an update on $\boldsymbol{x}$ may lead to a breakpoint where the condition (29) is violated. The first scenario occurs when an element of $\boldsymbol{c}$ not in the support set would increase in magnitude beyond λ:

$$\gamma^+ = \min_{i \notin \mathcal{I}} \left\{ \frac{\lambda - c_i}{1 - \boldsymbol{a}_i^T A_{\mathcal{I}} \boldsymbol{d}^{(k)}(\mathcal{I})}, \frac{\lambda + c_i}{1 + \boldsymbol{a}_i^T A_{\mathcal{I}} \boldsymbol{d}^{(k)}(\mathcal{I})} \right\}. \tag{32}$$

The index that achieves γ^+ is denoted as i^+. The second scenario occurs when an element of $\boldsymbol{c}$ in the support set $\mathcal{I}$ crosses zero, violating the sign agreement:

$$\gamma^- = \min_{i \in \mathcal{I}} \{-x_i / d_i\}. \tag{33}$$

The index that achieves γ^- is denoted as i^-. Hence, the homotopy algorithm marches to the next breakpoint, and updates the sparse support set by either appending $\mathcal{I}$ with i^+ or removing i^-:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \min\{\gamma^+, \gamma^-\} \boldsymbol{d}^{(k)}. \tag{34}$$

The algorithm terminates when the relative change in $\boldsymbol{x}$ between consecutive iterations is sufficiently small. Algorithm 2 summarizes an implementation of the Homotopy methods.[6]

Algorithm 2 Homotopy

Input: A full rank matrix $A = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n] \in \mathbb{R}^{d \times n}$, $d < n$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, initial Lagrangian parameter $\lambda = 2\|A^T \boldsymbol{b}\|_\infty$.

1: Initialization: $k \leftarrow 0$. Find the first support index: $i = \arg\max_{j=1}^n \|\boldsymbol{v}_j^T \boldsymbol{b}\|$, $\mathcal{I} = \{i\}$.
2: **repeat**
3: $k \leftarrow k + 1$.
4: Solve for the update direction $\boldsymbol{d}^{(k)}$ in (31).
5: Compute the sparse support updates (32) and (33): $\gamma^* \leftarrow \min\{\gamma^+, \gamma^-\}$.
6: Update $\boldsymbol{x}^{(k)}$, $\mathcal{I}$, and $\lambda \leftarrow \lambda - \gamma^*$.
7: **until** stopping criterion is satisfied.

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

Overall, solving (31) using a Cholesky factorization and the addition/removal of the sparse support elements dominate the computation. Since one can keep track of the rank-1 update of $A_{\mathcal{I}}^T A_{\mathcal{I}}$ in solving (31) using $O(d^2)$ operations in each iteration, the computational complexity of the homotopy algorithm is $O(kd^2 + kdn)$.

[6] A MATLAB implementation [1] can be found at `http://users.ece.gatech.edu/~sasif/homotopy/`.

Finally, we want to point out that Homotopy has been shown to share some connections with two greedy ℓ_1-min approximations, namely, *least angle regression* (LARS) [22] and *polytope faces pursuit* (PFP) [42]. For instance, if the ground-truth signal $\boldsymbol{x}_0$ has at most k non-zero components with $k << n$, all three algorithms can recover it in k iterations. On the other hand, LARS never removes indices from the sparse support set during the iteration, while Homotopy and PFP have mechanisms to remove coefficients from the sparse support. More importantly, Homotopy provably solves ℓ_1-min (P_1), while LARS and PFP are only approximate solutions. A more detailed comparison between Homotopy, LARS, and PFP can be found in [21].

2.3 Iterative Shrinkage-Thresholding Methods

Although Homotopy employs a more efficient iterative update rule that only involves operations on those submatrices of A corresponding to the support sets of $\boldsymbol{x}$, it may not be as efficient when the sparsity k and the observation dimension d grow proportionally with the signal dimension n. In such scenarios, one can show that the worst-case computational complexity is still bounded by $O(n^3)$. In this section, we discuss *Iterative Shrinkage-Thresholding* (IST) methods [16, 14, 26, 50], whose implementation mainly involves simple operations such as vector algebra and matrix-vector multiplications. This is in contrast to most past methods that all involve expensive operations such as matrix factorization and solving linear least squares problems. A short survey on the applications of IST can be found in [54].

In a nutshell, IST considers solving ($P_{1,2}$) as a special case of the following *composite objective function*:

$$\min_x F(\boldsymbol{x}) \doteq f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}), \tag{35}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth and convex function, and $g : \mathbb{R}^n \to \mathbb{R}$ as the regularization term is bounded from below but not necessarily smooth nor convex. For ℓ_1-min in particular, g is also separable, that is,

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} g_i(x_i). \tag{36}$$

Clearly, let $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2$ and $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$. Then the objective function (35) becomes the unconstrained BPDN problem.

The update rule to minimize (35) is computed using a second-order approximation of f [50, 5]:

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \arg\min_{\boldsymbol{x}}\{f(\boldsymbol{x}^{(k)}) + (\boldsymbol{x} - \boldsymbol{x}^{(k)})^T \nabla f(\boldsymbol{x}^{(k)}) \\
&\quad + \tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_2^2 \cdot \nabla^2 f(\boldsymbol{x}^{(k)}) + \lambda g(\boldsymbol{x})\} \\
&\approx \arg\min_{\boldsymbol{x}}\{(\boldsymbol{x} - \boldsymbol{x}^{(k)})^T \nabla f(\boldsymbol{x}^{(k)}) \\
&\quad + \tfrac{\alpha^{(k)}}{2}\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_2^2 + \lambda g(\boldsymbol{x})\} \\
&= \arg\min_{\boldsymbol{x}}\{\tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{u}^{(k)}\|_2^2 + \tfrac{\lambda}{\alpha^{(k)}} g(\boldsymbol{x})\}, \\
&\doteq G_{\alpha^{(k)}}(\boldsymbol{x}^{(k)}),
\end{aligned} \tag{37}$$

where

$$\boldsymbol{u}^{(k)} = \boldsymbol{x}^{(k)} - \frac{1}{\alpha^{(k)}} \nabla f(\boldsymbol{x}^{(k)}). \tag{38}$$

In (37), the Hessian $\nabla^2 f(\boldsymbol{x}^{(k)})$ is approximated by a diagonal matrix $\alpha^{(k)} I$.

If we replace $g(\boldsymbol{x})$ in (37) by the ℓ_1-norm $\|\boldsymbol{x}\|_1$, which is a separable function, then $G_{\alpha^{(k)}}(\boldsymbol{x}^{(k)})$ has a closed-form solution w.r.t. each component:

$$\begin{aligned} x_i^{(k+1)} &= \arg\min_{\boldsymbol{x}_i} \left\{ \tfrac{(x_i - u_i^{(k)})^2}{2} + \tfrac{\lambda |x_i|}{\alpha^{(k)}} \right\} \\ &= \operatorname{soft}\left(u_i^{(k)}, \tfrac{\lambda}{\alpha^{(k)}} \right), \end{aligned} \tag{39}$$

where

$$\begin{aligned} \operatorname{soft}(u, a) &\doteq \operatorname{sgn}(u) \max\{|u| - a, 0\} \\ &= \begin{cases} \operatorname{sgn}(u)(|u| - a) & \text{if } |u| > a \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{40}$$

is the *soft-thresholding* or *shrinkage* function [18].

There are two free parameters in (37), namely, the regularizing coefficient λ and the coefficient $\alpha^{(k)}$ that approximates the hessian matrix $\nabla^2 f$. Different strategies for choosing these parameters have been proposed. Since αI mimics the Hessian $\nabla^2 f$, we require that $\alpha^{(k)}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}) \approx \nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})$ in the least-squares sense. Hence,

$$\begin{aligned} \alpha^{(k+1)} &= \arg\min_{\alpha} \|\alpha(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}) \\ &\quad - \left(\nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})\right) \|_2^2 \\ &= \frac{\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)^T \left(\nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})\right)}{\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)^T \left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)}. \end{aligned} \tag{41}$$

This is known as the *Barzilai-Borwein* equation [3, 43, 50].

For choosing λ, instead of using a fixed value, several papers have proposed a *continuation* strategy [26, 23], in which (37) is solved for a decreasing sequence of λ. As mentioned in Section 2.2, (37) recovers the optimal ℓ_1-min solution when $\lambda \to 0$. However, it has been observed that in practice the performance degrades by directly solving (37) for small values of λ, which has been dubbed as a "cold" starting point. Instead, *continuation* employs a warm-starting strategy by first solving (37) for a larger value of λ, then decreasing λ in steps towards its desired value.

The IST algorithm is summarized in Algorithm 3.[7] The interested reader is referred to [33] for guidelines on choosing good soft-thresholding parameters to improve accuracy.

2.4 Proximal Gradient Methods

Proximal Gradient (PG) algorithms represent another class of algorithms that solve convex optimization problems of the form (35). Assume that our cost

[7] A MATLAB implementation called *Sparse Reconstruction by Separable Approximation* (SpaRSA) [50] is available at `http://www.lx.it.pt/~mtf/SpaRSA/`.

Algorithm 3 Iterative Shrinkage-Thresholding (IST)

Input: A full rank matrix $A \in \mathbb{R}^{d\times n}$, $d < n$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, Lagrangian λ_0, initial values for $\boldsymbol{x}^{(0)}$ and α^0, $k \leftarrow 0$.

1: Generate a reducing sequence $\lambda_0 > \lambda_1 > \cdots > \lambda_N$.
2: **for** $i = 0, 1, \cdots, N$ **do**
3: $\quad\lambda \leftarrow \lambda_i$
4: $\quad$**repeat**
5: $\quad\quad k \leftarrow k + 1$.
6: $\quad\quad \boldsymbol{x}^{(k)} \leftarrow G(\boldsymbol{x}^{(k-1)})$.
7: $\quad\quad$ Update $\alpha^{(k)}$ using (41).
8: $\quad$**until** The objective function $F(\boldsymbol{x}^{(k)})$ decreases.
9: **end for**

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

function $F(\cdot)$ can be decomposed as the sum of two functions f and g, where f is a smooth convex function with Lipschitz continuous gradient, and g is a continuous convex function. The principle behind these algorithms is to iteratively form quadratic approximations $Q(\boldsymbol{x}, \boldsymbol{y})$ to $F(\boldsymbol{x})$ around a carefully chosen point $\boldsymbol{y}$, and to minimize $Q(\boldsymbol{x}, \boldsymbol{y})$ rather than the original cost function F.

For our problem, we define $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ and $f(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$. We note that $\nabla f(\boldsymbol{x}) = A^T(A\boldsymbol{x} - \boldsymbol{b})$ is Lipschitz continuous with Lipschitz constant $L_f \doteq \|A\|^2$. Define $Q(\boldsymbol{x}, \boldsymbol{y})$ as:

$$Q(\boldsymbol{x}, \boldsymbol{y}) \doteq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{L_f}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\, g(\boldsymbol{x}). \tag{42}$$

It can be shown that $F(\boldsymbol{x}) \leq Q(\boldsymbol{x}, \boldsymbol{y})$ for all $\boldsymbol{y}$, and

$$\arg\min_{\boldsymbol{x}} Q(\boldsymbol{x}, \boldsymbol{y}) = \arg\min_{\boldsymbol{x}} \left\{ \lambda g(\boldsymbol{x}) + \frac{L_f}{2}\|\boldsymbol{x} - \boldsymbol{u}\|^2 \right\}, \tag{43}$$

where $\boldsymbol{u} = \boldsymbol{y} - \frac{1}{L_f}\nabla f(\boldsymbol{y})$ by the same trick used in (37). For the ℓ_1-min problem, (43) has a closed-form solution given by the soft-thresholding function:

$$\arg\min_{\boldsymbol{x}} Q(\boldsymbol{x}, \boldsymbol{y}) = \text{soft}\left(\boldsymbol{u}, \frac{\lambda}{L_f}\right). \tag{44}$$

However, unlike the iterative thresholding algorithm described earlier, we use a smoothed computation of the sequence $\boldsymbol{y}_k$. It has been shown that choosing

$$\boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)} + \frac{t_{k-1} - 1}{t_k}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right), \tag{45}$$

where $\{t_k\}$ is a positive real sequence satisfying $t_k^2 - t_k \leq t_{k-1}^2$, achieves an accelerated non-asymptotic convergence rate of $O(k^{-2})$ [38, 39, 5]. To further

accelerate the convergence of the algorithm, one can also make use of the continuation technique described in Section 2.3.

Finally, for large problems, it is often computationally expensive to directly compute $L_f = \|A\|^2$.[8] A *backtracking* line-search strategy [5] can be used to generate a scalar sequence $\{L_k\}$ that approximates L_f. We define

$$Q_L(\boldsymbol{x}, \boldsymbol{y}) \doteq f(\boldsymbol{y}) + (\boldsymbol{x} - \boldsymbol{y})^T \nabla f(\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\, g(\boldsymbol{x}). \tag{46}$$

Suppose that $\eta > 1$ is a pre-defined constant. Then, given $\boldsymbol{y}^{(k)}$ at the kth iteration, we set $L_k = \eta^j\, L_{k-1}$, where j is the smallest nonnegative integer such that the following inequality holds:

$$F(G_{L_k}(\boldsymbol{y}^{(k)})) \le Q_{L_k}(G_{L_k}(\boldsymbol{y}^{(k)}), \boldsymbol{y}^{(k)}), \tag{47}$$

where $G_L(\boldsymbol{y}) \doteq \arg\min_{\boldsymbol{x}}\, Q_L(\boldsymbol{x}, \boldsymbol{y}) = \text{soft}\left(\boldsymbol{u}, \frac{\lambda}{L}\right)$ for $\boldsymbol{u} \doteq \boldsymbol{y} - \frac{1}{L}\nabla f(\boldsymbol{y})$.

The algorithm, dubbed FISTA in [5], is summarized as Algorithm 4.[9] The convergence behavior of FISTA is given by

$$F(\boldsymbol{x}^{(k)}) - F(\boldsymbol{x}^*) \le \frac{2L_f\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2}{(k+1)^2}, \quad \forall\, k. \tag{48}$$

The interested reader may refer to [39, 5, 6] for a proof of the above result.

Algorithm 4 Fast Iterative Shrinkage-Threshold Algorithm (FISTA)

Input: $\boldsymbol{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$.

Set $\boldsymbol{x}^{(0)} \leftarrow 0$, $\boldsymbol{x}^{(1)} \leftarrow 0$, $t_0 \leftarrow 1$, $t_1 \leftarrow 1$, $k \leftarrow 1$.
Initialize L_0, λ_1, $\beta \in (0, 1)$, $\bar{\lambda} > 0$.
while not converged **do**
  $\boldsymbol{y}^{(k)} \leftarrow \boldsymbol{x}^{(k)} + \frac{t_{k-1}-1}{t_k}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)$.
  Update L_k using (47) with $\boldsymbol{y}^{(k)}$.
  $\boldsymbol{u}^{(k)} \leftarrow \boldsymbol{y}^{(k)} - \frac{1}{L_k}\, A^T(A\boldsymbol{y}^{(k)} - \boldsymbol{b})$.
  $\boldsymbol{x}^{(k+1)} \leftarrow \text{soft}\left(\boldsymbol{u}^{(k)}, \frac{\lambda_k}{L_k}\right)$.
  $t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$.
  $\lambda_{k+1} \leftarrow \max(\beta\,\lambda_k, \bar{\lambda})$.
  $k \leftarrow k + 1$.
end while

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

2.5 Augmented Lagrange Multiplier Methods

Lagrange multiplier methods are a popular class of algorithms in convex programming. The basic idea is to eliminate equality constraints and instead add

[8] This problem occurs in the IST algorithm as well.

[9] An implementation of FISTA is provided on the website of this paper. Another Matlab toolbox called NESTA [6] is available at: `http://www.acm.caltech.edu/~nesta/`.

a penalty term to the cost function that assigns a very high cost to infeasible points. Augmented Lagrange Multiplier (ALM) methods differ from other penalty-based approaches by simultaneously estimating the optimal solution and Lagrange multipliers in an iterative fashion.

We consider the general ℓ_1-min problem (1) with the optimal solution $\boldsymbol{x}^*$. The corresponding augmented Lagrange function is given by

$$L_\mu(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|_1 + \langle \boldsymbol{y}, \boldsymbol{b} - A\boldsymbol{x} \rangle + \frac{\mu}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2, \tag{49}$$

where $\mu > 0$ is a constant that determines the penalty for infeasibility, and $\boldsymbol{y}$ is a vector of Lagrange multipliers. Let $\boldsymbol{y}^*$ be a Lagrange multiplier vector satisfying the second-order sufficiency conditions for optimality (see section 3.2 in [8] for more details). Then, for sufficiently large μ, it can be shown that

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \; L_\mu(\boldsymbol{x}, \boldsymbol{y}^*). \tag{50}$$

The main problem with this solution is that $\boldsymbol{y}^*$ is unknown in general. Furthermore, the choice of μ is not straightforward from the above formulation. It is clear that to compute $\boldsymbol{x}^*$ by minimizing $L_\mu(\boldsymbol{x}, \boldsymbol{y})$, we must choose $\boldsymbol{y}$ close to $\boldsymbol{y}^*$ and set μ to be a very large positive constant. The following iterative procedure has been proposed in [8] to simultaneously compute $\boldsymbol{y}^*$ and $\boldsymbol{x}^*$:

$$\begin{cases} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \; L_{\mu_k}(\boldsymbol{x}, \boldsymbol{y}_k) \\ \boldsymbol{y}_{k+1} &= \boldsymbol{y}_k + \mu_k(\boldsymbol{b} - A\boldsymbol{x}_{k+1}) \end{cases}, \tag{51}$$

where $\{\mu_k\}$ is a monotonically increasing positive sequence. We note that the first step in the above procedure is itself an unconstrained convex optimization problem. Thus, the above iterative procedure is computationally efficient only if it is easier to minimize the augmented Lagrangian function compared to solving the the original constrained optimization problem (1) directly.

We focus our attention on solving the first step of (51) for the ℓ_1-min problem. Although it is not possible to obtain a closed-form solution, the cost function has the same form as described in (35). Furthermore, the quadratic penalty term is smooth and has a Lipschitz continuous gradient. Therefore, we can solve it efficiently using proximal gradient methods (e.g., FISTA) described in Section 2.4.

The entire algorithm is summarized as Algorithm 5, where τ represents the largest eigenvalue of the matrix $A^T A$, and $\rho > 1$ is a constant. Although the steps in Algorithm 5 closely resemble those of Algorithm 4, we will later see that ALM in general is more accurate and converges faster.

We would like to point out that a similar algorithm called Alternating Directions Method (ADM) [53] essentially shares the same idea as above.[10] The major difference is that ADM would approximate the solution to the first step of the ALM iteration in (51). This is done by computing only one iteration of

[10] A MATLAB toolbox of the ADM algorithm called YALL1 is provided at: `http://yall1.blogs.rice.edu/`.

Algorithm 5 Augmented Lagrange Multiplier (ALM)

Input: $\boldsymbol{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$.

while not converged ($k = 1, 2, \ldots$) **do**
  $t_1 \leftarrow 1$, $\boldsymbol{z}_1 \leftarrow \boldsymbol{x}_k$, $\boldsymbol{u}_1 \leftarrow \boldsymbol{x}_k$
  while not converged ($l = 1, 2, \ldots$) **do**
    $\boldsymbol{u}_{l+1} \leftarrow \text{soft}\left(\boldsymbol{z}_l - \frac{1}{\tau}A^T\left(A\boldsymbol{z}_l - \boldsymbol{b} - \frac{1}{\mu_k}\boldsymbol{y}_k\right), \frac{1}{\mu_k\tau}\right)$
    $t_{l+1} \leftarrow \frac{1}{2}\left(1 + \sqrt{1+4t_l^2}\right)$
    $\boldsymbol{z}_{l+1} \leftarrow \boldsymbol{u}_{l+1} + \frac{t_l - 1}{t_{l+1}}(\boldsymbol{u}_{l+1} - \boldsymbol{u}_l)$
  end while
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{u}_{l+1}$
  $\boldsymbol{y}_{k+1} \leftarrow \boldsymbol{y}_k + \mu_k(\boldsymbol{b} - A\boldsymbol{x}_{k+1})$
  $\mu_{k+1} \leftarrow \rho \cdot \mu_k$
end while

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}_k$.

the FISTA algorithm. Although this approximation yields a gain in computational speed, we have observed from experiments that the convergence behavior of ADM may vary depending on the distribution of the matrix A.

Algorithm 5 summarizes the ALM method applied to the primal problem (1), which is referred to as Primal ALM (PALM) in this paper. Interestingly, the principles of ALM can also be applied to its dual problem [53]:

$$\max_{\boldsymbol{y}} \boldsymbol{b}^T\boldsymbol{y} \quad \text{subj. to} \quad A^T\boldsymbol{y} \in \boldsymbol{B}_1^\infty, \tag{52}$$

where $\boldsymbol{B}_1^\infty = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_\infty \leq 1\}$. Under certain circumstances, this may result in a more computationally efficient algorithm. In the rest of the section, we will briefly explain the Dual ALM (DALM) algorithm.

Simple computation shows that DALM solves the following problem:

$$\begin{array}{ll} \min_{\boldsymbol{y},\boldsymbol{x},\boldsymbol{z}} & -\boldsymbol{b}^T\boldsymbol{y} - \boldsymbol{x}^T(\boldsymbol{z} - A^T\boldsymbol{y}) + \frac{\beta}{2}\|\boldsymbol{z} - A^T\boldsymbol{y}\|_2^2 \\ \text{subj. to} & \boldsymbol{z} \in \boldsymbol{B}_1^\infty. \end{array} \tag{53}$$

Here, $\boldsymbol{x}$ as the primal variable becomes the associated Lagrange multiplier in the dual space [53]. Since it is difficult to solve the above problem simultaneously w.r.t. $\boldsymbol{y}$, $\boldsymbol{x}$ and $\boldsymbol{z}$, we adopt a strategy by alternately updating the primal variable $\boldsymbol{x}$ and the dual variables $\boldsymbol{y}$ and $\boldsymbol{z}$.

On one hand, for $\boldsymbol{x} = \boldsymbol{x}_k$ and $\boldsymbol{y} = \boldsymbol{y}_k$, the minimizer $\boldsymbol{z}_{k+1}$ with respect to $\boldsymbol{z}$ is given by

$$\boldsymbol{z}_{k+1} = \mathcal{P}_{\boldsymbol{B}_1^\infty}(A^T\boldsymbol{y}_k + \boldsymbol{x}_k/\beta), \tag{54}$$

where $\mathcal{P}_{\boldsymbol{B}_1^\infty}$ represents the projection operator onto $\boldsymbol{B}_1^\infty$. On the other hand, given $\boldsymbol{x} = \boldsymbol{x}_k$ and $\boldsymbol{z} = \boldsymbol{z}_{k+1}$, the minimization with respect to $\boldsymbol{y}$ is a least squares problem, whose solution is given by the solution to the following equation:

$$\beta AA^T\boldsymbol{y} = \beta A\boldsymbol{z}_{k+1} - (A\boldsymbol{x}_k - \boldsymbol{b}). \tag{55}$$

Suppose that AA^T is invertible. Then, we directly use its inverse to solve (55). However, for large scale problems, this matrix inversion can be computationally expensive. Therefore, in such cases, we can approximate the solution with one step of the Conjugate Gradient algorithm in the $\boldsymbol{y}$ direction at each iteration, as proposed in [53].

The basic iteration of the DALM algorithm is given by[11]

$$\begin{cases} \boldsymbol{z}_{k+1} = \mathcal{P}_{\boldsymbol{A}_1^\infty}(A^T\boldsymbol{y}_k + \boldsymbol{x}_k/\beta), \\ \boldsymbol{y}_{k+1} = (AA^T)^{-1}(A\boldsymbol{z}_{k+1} - (A\boldsymbol{x}_k - \boldsymbol{b})/\beta), \\ \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \beta(\boldsymbol{z}_{k+1} - A^T\boldsymbol{y}_{k+1}), \end{cases} \tag{56}$$

As we will see in Section 4, when the matrix AA^T is easily invertible, it is possible to solve (55) exactly. Since all the subproblems now are solved exactly, the convergence of the dual algorithm is guaranteed. Furthermore, since its major computation lies in solving for the dual variable $\boldsymbol{y}$, when the number of dual variables are much smaller than the number of primal variables (i.e., when $d \ll n$), the dual algorithm could be more efficient than the primal algorithm.

3 Simulation: Random Sparse Signals

In this section, we present two sets of experiment to benchmark the performance of the five fast ℓ_1-min algorithms on random sparse signals, namely, L1LS/TNIPM, Homotopy, SpaRSA/IST, FISTA, and DALM, together with the basic BP algorithm (i.e., PDIPA). All experiments are performed in MATLAB on a Dell PowerEdge 1900 workstation with dual quad-core 2.66GHz Xeon processors and 8GB of memory.

Before we present the results, we need to caution that the performance of these algorithms is affected by multiple factors, including the algorithm implementation, the chosen ad-hoc parameters, and even the MATLAB environment itself. One factor that we should pay special attention to is the stopping criteria used in benchmarking these algorithms. As we first mentioned in Section 1.2, choosing a good stopping criterion is important to properly exit an iteration when the estimate becomes close to a local or global optimum. On one hand, in general, straightforward rules do exist, such as the relative change of the objective function:

$$\frac{\|F(\boldsymbol{x}^{(k+1)}) - F(\boldsymbol{x}^{(k)})\|}{\|F(\boldsymbol{x}^{(k)})\|}, \tag{57}$$

or the relative change of the estimate:

$$\frac{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^{(k)}\|}. \tag{58}$$

However, their efficacy depends on a proper step size of the update rule: If the step size is poorly chosen, the algorithm may terminate prematurely when the

[11] The PALM and DALM algorithms in MATLAB can be downloaded from the website of this paper.

estimate is still far away from the optimum. On the other hand, certain special criteria are more effective to some algorithms than the others. For example, for PDIPA, it is natural to use the (relative) duality gap between the primal and dual solutions; for Homotopy, it is easy to measure the change of the sparse support as the stopping criterion.

In this section, since the experiment is performed on synthetic data, we use a simple threshold that compares the ℓ_2-norm difference between the ℓ_1-min estimate $\boldsymbol{x}^*$ and the ground truth $\boldsymbol{x}_0$ as the stopping criterion.

3.1 ρ-δ Plot in the Noise-Free Case

The first experiment is designed to measure the accuracy of the various algorithms in recovering sparse signals in the noise-free case (P_1). We evaluate each algorithm using a ρ-δ plot, where the sparsity rate $\rho = k/n \in (0, 1]$ and the sampling rate $\delta = d/n \in (0, 1]$. At each δ, the percentages of successes that an ℓ_1-min algorithm finds the ground-truth solution $\boldsymbol{x}_0$ (with a very small tolerance threshold) are measured over different ρ's. Then a fixed success rate, say of 95%, over all δ's can be interpolated as a curve in the ρ-δ plot. In general, the higher the success rates, the better an algorithm can recover dense signals in the (P_1) problem.

Figure 2 shows the 95% success-rate curves for the six algorithms. In the simulation, the ambient dimension $d = 1000$ is fixed. For a fixed pair (ρ, δ), the support of the ground-truth signal $\boldsymbol{x}_0$ is chosen uniformly at random, and the values of the non-zero entries are drawn from the standard normal distribution. In addition, the vector is normalized to have unit ℓ_2-norm. The measurement matrix A is generated as a random Gaussian matrix, each of whose entries is i.i.d. randomly generated from the standard normal distribution and then normalized to have unit column norm. We choose to compute the average success rates over 100 trials on the vertices of a grid of (ρ, δ) pairs for each of the ℓ_1-min algorithms, and the coordinates of the 95% rate are interpolated from the vertex values.

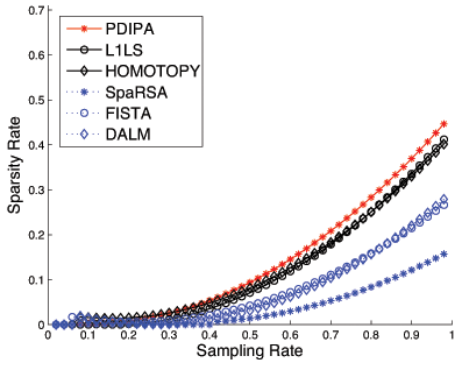


Fig. 2: The ρ-δ plot (in color) shows the 95% success-rate curves for the six fast ℓ_1-min algorithms.

The observations of the experiment are summarized below:

1. Without concerns about speed and data noise, the success rates of the interior-point method PDIPA is the highest of all the algorithms compared in Figure 2, especially when the signal becomes dense.

2. The accuracy for the remaining five algorithm is separated in three groups. In particular, the success rates of L1LS and Homotopy are similar w.r.t. different sparsity rates and sampling rates, and they are also very close to PDIPA. In the low sampling-rate regime, Homotopy is slightly better than L1LS.

3. The success rates of FISTA and DALM are comparable over all sampling rates. In the noise-free case, they are not as accurate as the other exact ℓ_1-min solutions. However, their performance shows a significant improvement over the IST algorithm, namely, SpaRSA.

3.2 Performance with Moderate Data Noise

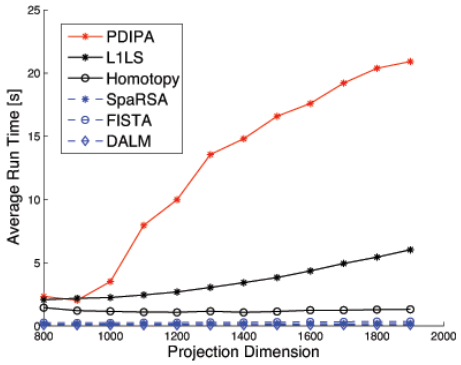
We are more interested in comparing the ℓ_1-min algorithms when the measurement contains moderate amounts of data noise. In the second simulation, we rank the six algorithms under two scenarios: Firstly, we measure the performance in the low-sparsity regime, where the ambient dimension $n = 2000$ and the sparsity ratio $\rho \doteq k/n = 0.1$ are fixed, and the dimension d of the Gaussian random projection varies from 800 to 1900. Secondly, we measure the performance when $\boldsymbol{x}$ becomes dense w.r.t. a fixed sampling rate, where $n = 2000$ and $d = 1500$ are fixed, and the sparsity ratio $\rho = k/n$ varies from 0.1 to 0.26. The maximum number of iterations for all algorithms is limited to 5000. The results are shown in Figure 3 and 4, respectively. In both experiments, we corrupt the measurement vector $\boldsymbol{b}$ with $\boldsymbol{e}$, an additive white noise term whose entries are i.i.d. according to a Gaussian distribution with zero mean and variance 0.01.

Firstly, when a low sparsity ratio of $\rho = 0.1$ is fixed in Figure 3, ℓ_1-min becomes better conditioned as the projection dimension increase. However, the computation cost also increases with the projection dimension. We then compare the performance of the six algorithm in Figure 3:

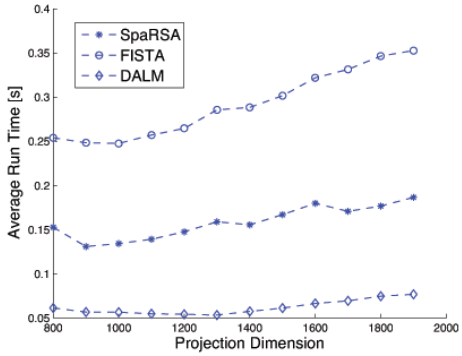
1. The computational complexity of PDIPA grows much faster than the other fast algorithms. More importantly, in contrast to its performance in the noise-free case, the estimation error also grows exponentially, in which case the algorithm (i.e., the update rule (11)) fails to converge to an estimate that is close to the ground truth.

2. As the projection dimension increases, the five fast ℓ_1-min algorithms all converge to good approximate solutions. The estimation error of Homotopy is slightly higher than the rest four algorithms.

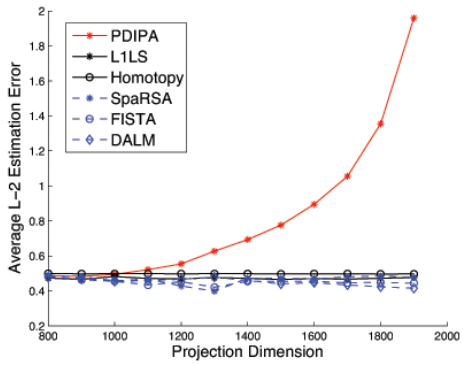
3. In terms of speed, L1LS and Homotopy take much longer time to converge than SpaRSA, FISTA, and DALM.



(a) Average run time

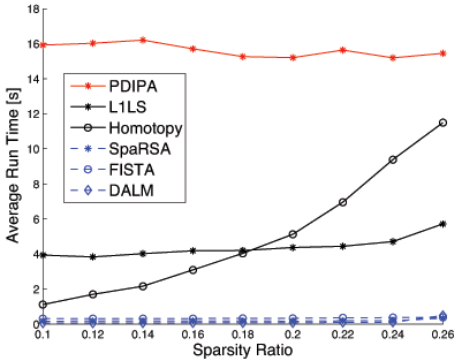


(b) Average run time (detail)

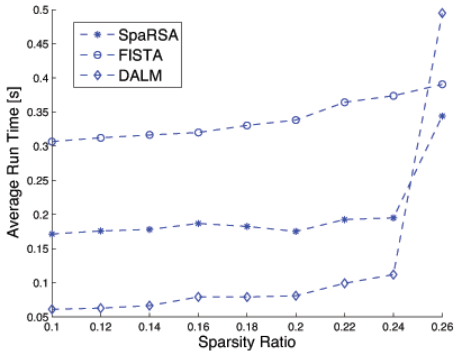


(c) Average ℓ_2-norm error

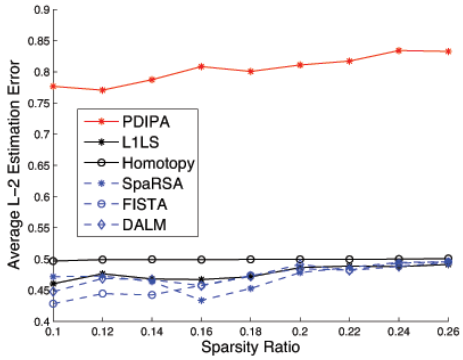
Fig. 3: Comparison of the six fast ℓ_1-min algorithms w.r.t. a fixed sparsity ratio ($n = 2000, k = 200$), and varying projection dimension d from 800 to 1900.



(a) Average run time



(b) Average run time (detail)



(c) Average ℓ_2-norm error

Fig. 4: Comparison of the six fast ℓ_1-min algorithms w.r.t. a fixed sampling ratio ($n = 2000, d = 1500$), and varying sparsity ratio k/n from 0.1 to 0.26.

4. The average run time of DALM is the shortest over all projection dimensions, which makes it the best algorithm in this comparison.

Secondly, when the projection dimension $d = 1500$ is fixed in Figure 4, we compare both the average run time and the average estimation error when the sparsity varies:

1. PDIPA significantly underperforms the rest five fast algorithms in terms of both accuracy and speed, consistent with the result in the previous simulation with noisy data.

2. The average run time of Homotopy grows almost linearly with the sparsity ratio, while the other algorithms are relatively unaffected. Thus, Homotopy is more suitable for scenarios where the unknown signal is expected to have a very small sparsity ratio.

3. The computational cost of the rest four algorithms, namely, L1LS, SpaRSA, FISTA, and DALM, remains largely unchanged when the random projection dimension d is fixed.

4. DALM is the fastest algorithm in the low-sparsity regime, but its run time approaches that of the other first-order methods in the high-sparsity regime. Overall, DALM is the best algorithm in this scenario.

To summarize, in the presence of low levels of Gaussian noise, PDIPA performs quite badly in comparison with the other ℓ_1-min algorithms that employ more sophisticated techniques to handle noisy data. Perhaps a more surprising observation is that when the coefficients of A are randomly drawn from a Gaussian distribution, under a broad range of simulation conditions, FISTA is *not* significantly better than the original IST algorithm.[12] However, in the next section, we will see that when the dictionary A is replaced by real-world data matrices that contain training images in face recognition, the performance of the six ℓ_1-min algorithms would be dramatically different from synthetic data.

4 Face Recognition Experiment I: Corruption and Disguise Compensation

In this section, we benchmark the performance of the six algorithms in robust face recognition. The experiment is set up to estimate sparse representation of real face images based on the *cross-and-bouquet* (CAB) model introduced in [47].

More specifically, It has been known in face recognition that a well-aligned frontal face image $\boldsymbol{b}$ under different lighting and expression lies close to a special low-dimensional linear subspace spanned by the training samples from the same subject, called a face subspace [7, 4]:

$$A_i = [\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \cdots, \boldsymbol{v}_{i,n_i}] \in \mathbb{R}^{d \times n_i}, \tag{59}$$

[12] We have observed that the performance of FISTA may vary widely depending on its chosen parameters. If the parameters are tuned at different noise level, its speed of convergence can be much improved compared to IST. However, for consistency, all algorithms in this simulation are assigned a fixed set of parameters.

where $\boldsymbol{v}_{i,j}$ represents the j-th training image from the i-th subject stacked in the vector form. Given C subjects and a new test image $\boldsymbol{b}$ (also in the vector form), we seek the sparsest linear representation of the sample with respect to all training examples:

$$\boldsymbol{b} = [A_1, A_2, \cdots, A_C][\boldsymbol{x}_1; \boldsymbol{x}_2; \cdots; \boldsymbol{x}_C] = A\boldsymbol{x}, \tag{60}$$

where $A \in \mathbb{R}^{d\times n}$ collects all the training images.

Clearly, if $\boldsymbol{b}$ is a valid test image, it must be associated with one of the C subjects. Therefore, the corresponding representation in (60) has a sparse representation $\boldsymbol{x} = [\cdots; \mathbf{0}; \boldsymbol{x}_i; \mathbf{0}; \cdots]$: on average only a fraction of $\frac{1}{C}$ coefficients are nonzero, and the dominant nonzero coefficients in sparse representation $\boldsymbol{x}$ reveal the true subject class.

In addition, we consider the situation where the query image $\boldsymbol{b}$ may be severely occluded or corrupted. The problem is modeled by a corrupted set of linear equations $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{e} \in \mathbb{R}^d$ is an unknown vector whose nonzero entries correspond to the corrupted pixels. In [49], the authors proposed to estimate $\boldsymbol{w} \doteq [\boldsymbol{x}; \boldsymbol{e}]$ jointly as the sparsest solution to the extended equation:

$$\min \|\boldsymbol{w}\|_1 \text{ subject to } \boldsymbol{b} = [A,\ I]\boldsymbol{w}. \tag{61}$$

The new dictionary $[A,\ I]$ was dubbed a cross-and-bouquet model in the following sense. The columns of A are highly correlated, as the convex hull spanned by all face images of all subjects occupies an extremely tiny portion of the ambient space. These vectors are tightly bundled together as a "bouquet," whereas the vectors associated with the identity matrix and its negative $\pm I$ form a standard "cross" in $\mathbb{R}^d$, as shown in Figure 5.

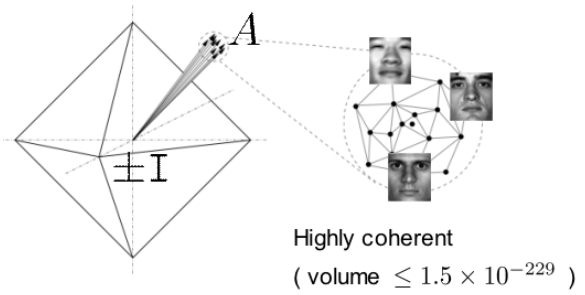


Fig. 5: The cross-and-bouquet model for face recognition. The raw images of human faces expressed as columns of A are clustered with very small variance. (Courtesy of John Wright [47])

In this section, the performance of the six ℓ_1-min algorithms using the CAB model is benchmarked on the CMU Multi-PIE face database [25]. A subset of 249 subjects from the data set (Session 1) is used for this experiment. Each subject is captured under 20 different illumination conditions with a frontal pose. The images are then manually aligned and cropped, and down-sampled to 40×30

pixels. Out of the 20 images for each subject, images $\{0, 1, 7, 13, 14, 16, 18\}$ with extreme illumination conditions are chosen as the training images, and the remaining 13 images are used as test images. Finally, a certain number of image pixels are randomly corrupted by a uniform distribution between $[0, 255]$, with the corruption percentage from 0% to 90%, as four examples shown in Figure 6. In Table 1, the recognition rates between 50% and 70% pixel corruption are highlighted for more detailed comparison.

Fig. 6: An aligned face image of Subject 1 in Multi-PIE, Session 1, under the ambient lighting condition (No. 0) is shown on the left. On the right, 20%, 40%, 60%, and 80% of image pixels are randomly selected and corrupted with uniformly distributed values in [0, 255], respectively.

Fig. 7: Recovered face images ($\hat{\boldsymbol{b}} = \boldsymbol{b} - \boldsymbol{e}$) from Figure 6 using the CAB model (61) and Homotopy. **Left:** Reconstructed image from 20% pixels corruption. **Middle Left:** Reconstructed image from 40% pixels corruption. **Middle Right:** Reconstructed image from 60% pixels corruption. **Right:** Reconstructed image from 80% pixels corruption.

We measure the performance of the algorithms in terms of the final recognition rate and the speed of execution. In choosing a proper stopping criterion, the stopping threshold is individually tuned for each algorithm to achieve the highest recognition rate. The results are shown in Figure 8. In addition, Figure 7 shows the reconstructed face images after corruption compensation, namely, $\hat{\boldsymbol{b}} = \boldsymbol{b} - \boldsymbol{e}$, given by the Homotopy solver.

Tab. 1: Average recognition rates (in percentage) between 50% and 70% random pixel corruption on the multi-PIE database. The highest rates are in boldface.

Corr.	PDIPA	L1LS	Homotopy	SpaRSA	FISTA	DALM
50%	**99.8**	99.5	**99.8**	97.6	96.2	99.5
60%	98.6	96.6	**98.7**	90.5	86.8	96.2
70%	84.1	76.3	**84.6**	63.3	58.7	78.8

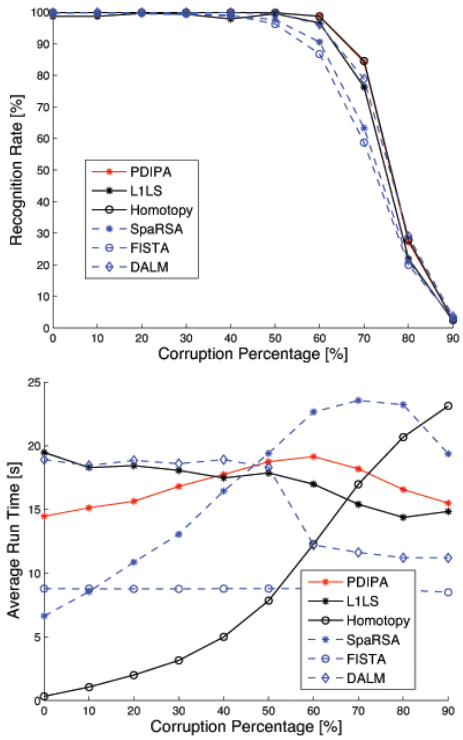


Fig. 8: **Left:** Average recognition accuracy (in percentage) on the Multi-PIE database. **Right:** Average run time (in second) on the Multi-PIE database.

In terms of accuracy, Homotopy achieves the best overall performance. For instance, with 60% of the pixels randomly corrupted, its average recognition rate based on the CAB model is about 99%. The performance of PDIPA is very close to Homotopy, achieving the second best overall accuracy. On the other hand, FISTA obtains the lowest recognition rates, followed by SpaRSA.

In terms of speed, Homotopy is also one of the fastest algorithm. In particular, when the pixel corruption percentage is small, the sparsity of the coefficient vector $\boldsymbol{w} = [\boldsymbol{x}; \boldsymbol{e}]$ is very small. As Homotopy iteratively adds or removes nonzero coefficients one at a time, the algorithm can quickly terminate after just a few iterations. On the other hand, as $\boldsymbol{w}$ becomes dense when the pixel corruption percentage increases, the complexity of Homotopy increases superlinearly and becomes inefficient. It is also important to note that the speed of the two accelerated iterative-thresholding algorithms, namely, FISTA and DALM, does not increase significantly as the sparsity of $\boldsymbol{w}$ increases.

It is more interesting to separately compare PDIPA, L1LS, and Homotopy, which provably solve the (P_1) problem, and SpaRSA, FISTA, and DALM, which essentially solve a relaxed version of (P_1). Overall, PDIPA, L1LS and Homotopy perform similarly in terms of the accuracy, consistent with our previous obser-

vations in the simulation. However, L1LS is significantly more expensive from a computation point of view to achieve the same recognition rates as Homotopy. Among the three iterative soft-thresholding methods, the experiments suggest that DALM is the most accurate.

5 Face Recognition Experiment II: Face Alignment

In this section, we benchmark the performance of the ℓ_1-min algorithms in robust face alignment within the sparse representation framework. While the previous section has demonstrated that solving the CAB model achieves state-of-the-art recognition accuracy on public datasets even when the test face images are severely corrupted or occluded, its success still relies on the assumption that the test images are well aligned with the training images. However, this assumption is not always satisfied in practice. As illustrated in Figure 9 top, a test face image obtained from an off-the-shelf face detector is likely to have some registration error against the training images. The registration error may contribute to erroneous linear representation of the query image, even if sufficient illuminations are present in the training set. Fortunately, this alignment issue can be naturally addressed within the sparse representation framework. It has been shown in [46] that the face alignment problem can be solved by a series of linear problems that iteratively minimize the sparsity of the registration error, which leads to an effective algorithm that works for face images under a large range of variations in scale change, 2-D translation and rotation, and different 3-D poses. Furthermore, it inherits the robustness property from the sparse representation framework, and hence is also able to deal with faces that are partially occluded.

Recall that a well-aligned test image $\boldsymbol{b}_0 \in \mathbb{R}^d$ can be represented as a sparse linear combination $A\boldsymbol{x}_0$ of all of the images in the database, plus a sparse error $\boldsymbol{e}_0$ due to occlusion: $\boldsymbol{b}_0 = A\boldsymbol{x}_0 + \boldsymbol{e}_0$. Now suppose that $\boldsymbol{b}_0$ is subject to some misalignment, so instead of observing $\boldsymbol{b}_0$, we observe the warped image

$$\boldsymbol{b} = \boldsymbol{b}_0 \circ \tau^{-1} \tag{62}$$

for some transformation $\tau \in T$, where T is a finite-dimensional group of transformations acting on the image domain. As seen in Figure 9, directly seeking a sparse representation of $\boldsymbol{b}$ against the (well-aligned) training images is no longer appropriate. However, if the true deformation τ^{-1} can be efficiently found, then we can apply its inverse τ to the test image, and it again becomes possible to find a good sparse representation of the resulting image:

$$\boldsymbol{b} \circ \tau = A\boldsymbol{x} + \boldsymbol{e}. \tag{63}$$

Naturally, we would like to use the sparsity as a cue for finding the correct deformation τ. This can be formulated as the following optimization problem:

$$\min_{\boldsymbol{x},\boldsymbol{e},\tau\in T} \|\boldsymbol{x}\|_1 + \|\boldsymbol{e}\|_1 \quad \text{subj to} \quad \boldsymbol{b} \circ \tau = A\boldsymbol{x} + \boldsymbol{e}. \tag{64}$$

However, this is a difficult nonconvex optimization problem. Furthermore, due to the concern of local minima, directly solving (64) may simultaneously align

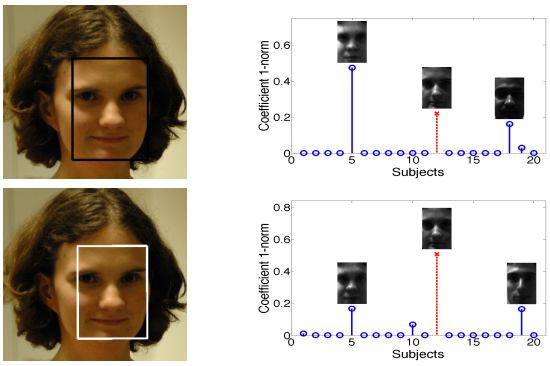


Fig. 9: **Effect of registration**. The task is to identify the girl among 20 subjects, by computing the sparse representation of her input face with respect to the entire training set. The absolute sum of the coefficients associated with each subject is plotted on the right. We also show the faces reconstructed with each subject's training images weighted by the associated sparse coefficients. The red line (cross) corresponds to her true identity, subject 12. **Top:** The face region is extracted by Viola and Jones' face detector (the black box). **Bottom:** Informative representation obtained by using a well-aligned face region (the white box). (Courtesy of [46]).

the test image to different subjects in the database. Therefore, it is more appropriate to seek the best alignment with each subject i in the database [46]:

$$\hat{\tau}_i = \arg \min_{\boldsymbol{x},\boldsymbol{e},\tau_i \in T} \|\boldsymbol{e}\|_1 \quad \text{subj to} \quad \boldsymbol{b} \circ \tau_i = A_i \boldsymbol{x} + \boldsymbol{e}. \tag{65}$$

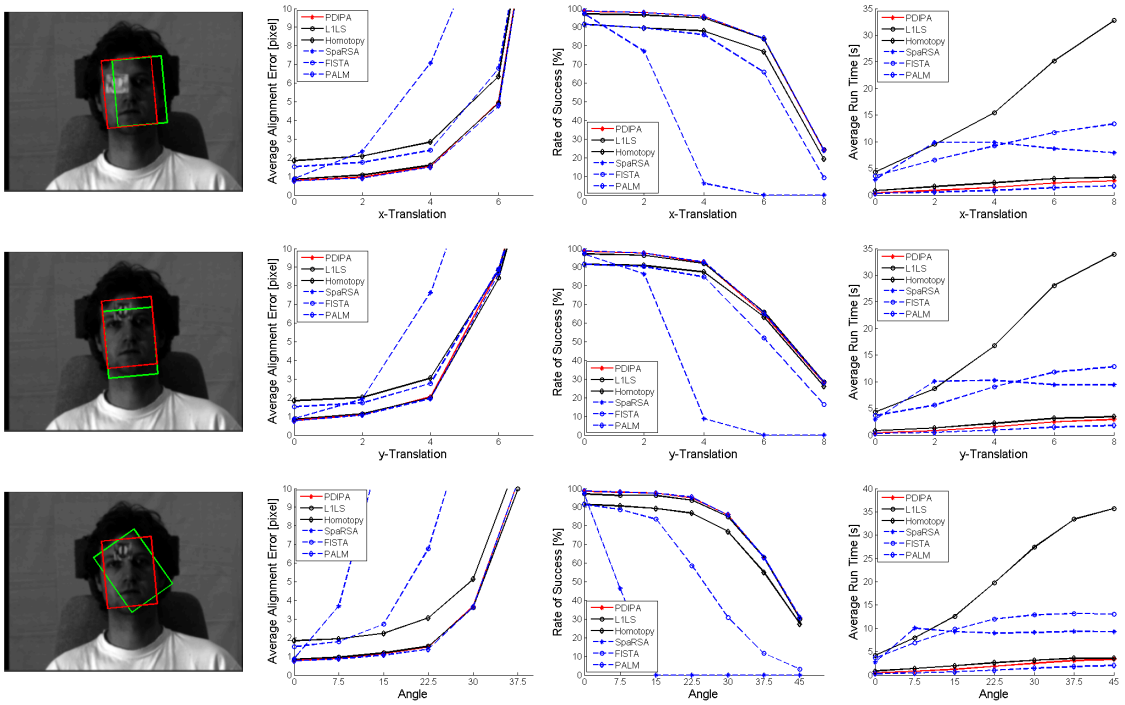
In (65), $\|\boldsymbol{x}\|_1$ is not penalized, since $A_i \in \mathbb{R}^{d \times n_i}$ only contains the images of subject i and $\boldsymbol{x}$ is no longer expected to be sparse. While (65) is still nonconvex, when a good initial estimation for the transformation is available, e.g., from the output of a face detector, one can refine this initialization to an estimate of the true transformation by repeatedly linearizing about the the current estimate of τ_i. This idea leads to the final problem formulation as a convex optimization problem as follows:

$$\min_{\boldsymbol{x},\boldsymbol{e},\Delta\tau_i \in T} \|\boldsymbol{e}\|_1 \quad \text{subj to} \quad \boldsymbol{b} \circ \tau_i + J_i \Delta\tau_i = A_i \boldsymbol{x} + \boldsymbol{e}. \tag{66}$$

Here, $J_i = \frac{\partial}{\partial \tau_i} \boldsymbol{b} \circ \tau_i \in \mathbb{R}^{d \times q_i}$ is the Jacobian of $\boldsymbol{b} \circ \tau_i$ with respect to the transformation parameters τ_i, and $\Delta\tau_i$ is the update direction w.r.t. τ_i.

During each iteration k, the current alignment parameters $\tau_i^{(k)}$ correct the observation as $b_i^{(k)} = b \circ \tau_i^{(k)}$. Denote $B_i^{(k)} = [A_i, -J_i^{(k)}] \in \mathbb{R}^{d \times (n_i + q_i)}$ and $\boldsymbol{w} = [\boldsymbol{x}^T, \Delta\tau_i^T]^T$, then the update $\Delta\tau_i$ can be computed by solving the following problem:

$$\min_{\boldsymbol{w},\boldsymbol{e}} \|\boldsymbol{e}\|_1 \quad \text{subj to} \quad \boldsymbol{b}_i^{(k)} = B_i^{(k)} \boldsymbol{w} + \boldsymbol{e}. \tag{67}$$



(a) (b) (c) (d)

Fig. 10: Alignment experiment results. (a) Examples of perturbation. The green bounding boxes indicate the initial face location with some registration error; the red bounding boxes show the final position after face alignment. (b) Average pixel error (lower the better). (c) Rate of Success (higher the better). (d) Average run time (lower the better).

Interested readers are referred to [46] for more details about the algorithm and comprehensive experimental studies.

In this section, we benchmark the various ℓ_1-min algorithms on the CMU Multi-PIE face database. In order to solve (67), we had to modify the code for each algorithm so that $\|\boldsymbol{x}\|_1$ is no longer penalized and to take care of the new constraint. To further speed up the algorithms, we take advantage of the fact that the matrix B is usually a tall matrix with much fewer columns than rows. For our experiments, the training set contains $n_i = 7$ images per subject, and we choose the transformation group T to be the set of similarity transforms (therefore $q_i = 4$). So, the number of columns in B is just $n_i + q_i = 11$, while the number of rows is equal to the number of pixels in each image.

Firstly, we modify the PDIPA to obtain a faster algorithm for this problem. We recall that PDIPA is computationally expensive mainly because at each iteration, we need to solve a linear system of equations the size of the dictionary $[B, I]$ for (67). For instance, if $d = 40 \times 30 = 1200$, it results in inverting a

matrix of size larger than 1200×1200. However, if we know that B has very few columns, it is not efficient to consider the dictionary $[B, I]$ as a whole. Instead, we can explicitly write down the closed-form solution to the large linear system with B and I considered separately. This leads to a much smaller problem that only requires inverting a $(n_i + q_i) \times (n_i + q_i) = 11 \times 11$ matrix. Hence, the complexity of PDIPA is significantly reduced. We note that similar ideas can be also applied to speed up the other algorithms. In particular, when matrix B is typically an overdetermined matrix, the ALM algorithm that operates in the primal space, namely, PALM, is more efficient than DALM in the dual space. In the Appendix, modifications for three ℓ_1-min algorithms are briefly explained, which solve (67) based on GP, Homotopy, and IST approaches, respectively.

The performance of the six ℓ_1-min algorithms is again evaluated using the face images from the Multi-PIE database. In this experiment, the 249 subjects from the Session 1 are used. Out of 20 illuminations, $\{0, 1, 7, 13, 14, 16, 18\}$ are chosen as the training images and the illumination 10 is used as the query image. All images are cropped and down-sampled to 40×30 pixels. Furthermore, the test images are manually perturbed up to 8 pixels along the x and y-axes in the canonical frame (with size 40×30) or up to 45° in-plane degree, respectively. Moreover, to test the robustness of the ℓ_1-min algorithms to occlusion, we replace a random located block of size 10% of the face image with the image of a baboon. See Figure 10(a) for an example.

The accuracy of each algorithm is measured by both the average pixel error and the rate of success. We consider an alignment successful if the difference between the final alignment is within 3 pixels of the ground truth in the original image frame (640×480 image size). As shown in Figure 10(b) and (c), in terms of alignment accuracy, PDIPA, PALM and L1LS achieve the best performance, with Homotopy and FISTA slightly worse. Meanwhile, the performance of SpaRSA degrades much faster than the others as the perturbation level increases. On the other hand, Figure 10(d) shows that L1LS, SpaRSA, and FISTA are more computational expensive. Overall, PALM is the fastest algorithm, which takes 25% to 50% less time compared to PDIPA and Homotopy.

In conclusion, Both PALM and PDIPA performs very well for the alignment problem. Taking into account the implementation complexity and the fact that PALM is slightly faster than PDIPA, we conclude that PALM is the best choice for this problem.

6 Conclusion and Discussion

We have provided a comprehensive review of five fast ℓ_1-min methods, i.e., *gradient projection*, *homotopy*, *soft shrinkage-thresholding*, *proximal gradient*, and *augmented Lagrange multiplier*. Through extensive experiments, we have shown that under a wide range of data conditions, there is no clear winner that always achieves the best performance in terms of both speed and accuracy. For perfect, noise-free data, the primal-dual interior point method (PDIPA) is more accurate than the other algorithms, albeit at a much lower speed. For noisy

data, first-order ℓ_1-min techniques (i.e., SpaRSA, FISTA, and ALM) are more efficient in recovering sparse signals in both the low-sparsity and high-sparsity regimes. We have also considered special cases of the ℓ_1-min problem where it is applied to robust face recognition. In this application, Homotopy and ALM achieve the highest recognition rates, and their computational cost is also the lowest compared to the other ℓ_1-min algorithms.

All experiments described in this paper were carried out in MATLAB. The benchmarking in terms of computational speed may vary significantly when the algorithms are implemented in other programming languages such as C/C++. The accuracy of different algorithms may also be affected by the precision of different floating-point/fixed-point arithmetic logic. To aid in peer evalution, all the experimental scripts and data have been made available on our website.

7 Appendix

In this section, we briefly discuss how to more efficiently solve the objective function (67) by taking advantage of the intrinsic structure of its dictionary $[B, I]$. For clarity, we will rewrite (67) using the following notation:

$$\min_{\boldsymbol{w},\boldsymbol{e}} \|\boldsymbol{e}\|_1 \quad \text{subj to} \quad \boldsymbol{b} = B\boldsymbol{w} + \boldsymbol{e}. \tag{68}$$

7.1 Solving Face Alignment via Gradient Projection Methods

The objective function (68) can be optimized relatively straightforward following the gradient projection approach. In particular, using *truncated Newton interior-point method*, the objective function is rewritten as:

$$\min \quad \tfrac{1}{2}\|\boldsymbol{b} - B\boldsymbol{w} - \boldsymbol{e}\|^2 + \lambda \textstyle\sum_{i=1}^{d} u_i \tag{69}$$

$$\text{subj. to} \quad -u_i \leq e_i \leq u_i, \quad i = 1, \cdots, d. \tag{70}$$

Then a logarithmic barrier to bound the domain of $-u_i \leq e_i \leq u_i$ can be constructed as:

$$\Phi(\boldsymbol{e}, \boldsymbol{u}) = -\sum \log(u_i + e_i) - \sum \log(u_i - e_i). \tag{71}$$

Over the domain of $(\boldsymbol{w}, \boldsymbol{e}, \boldsymbol{u})$, the central path is calculated by minimizing the following convex function

$$F_t(\boldsymbol{w}, \boldsymbol{e}, \boldsymbol{u}) = t(\frac{1}{2}\|\boldsymbol{b} - B\boldsymbol{w} - \boldsymbol{e}\|^2 + \lambda \sum_{i=1}^{d} u_i) + \Phi(\boldsymbol{e}, \boldsymbol{u}), \tag{72}$$

where t is a nonnegative parameter. Then, the update direction $(\Delta\boldsymbol{w}, \Delta\boldsymbol{e}, \Delta\boldsymbol{u})$ is solved using Newton's method:

$$\nabla^2 F_t(\boldsymbol{w}, \boldsymbol{e}, \boldsymbol{u}) \cdot \begin{bmatrix} \Delta\boldsymbol{w} \\ \Delta\boldsymbol{e} \\ \Delta\boldsymbol{u} \end{bmatrix} = -\nabla F_t(\boldsymbol{w}, \boldsymbol{e}, \boldsymbol{u}). \tag{73}$$

7.2 Solving Face Alignment via Homotopy Methods

In the Homotopy formulation, define a composite objective function:

$$F(\boldsymbol{w}, \boldsymbol{e}) = \frac{1}{2}\|\boldsymbol{b} - B\boldsymbol{w} - \boldsymbol{e}\|^2 + \lambda\|\boldsymbol{e}\|_1 \quad (74)$$

$$= f(\boldsymbol{w}, \boldsymbol{e}) + \lambda g(\boldsymbol{e}). \quad (75)$$

Setting the subgradient $\partial F = 0 \Leftrightarrow -\partial f = \lambda \partial g$, we obtain the following equalities:

$$\lambda \frac{\partial}{\partial \boldsymbol{w}} g(\boldsymbol{e}) = 0 = B^T(\boldsymbol{b} - B\boldsymbol{w} - \boldsymbol{e}) \quad (76)$$

and

$$\boldsymbol{c}(\boldsymbol{e}) \doteq \lambda \frac{\partial}{\partial \boldsymbol{e}} g(\boldsymbol{e}) = (\boldsymbol{b} - B\boldsymbol{w} - \boldsymbol{e}). \quad (77)$$

In initialization, set $\boldsymbol{e}^{(0)} = \boldsymbol{0}$ and $\boldsymbol{w} = B^{\dagger}\boldsymbol{b}$. Since we know $\partial\|\boldsymbol{e}\|_1 \in [-1, 1]$, we initialize the maximal λ using (77):

$$\lambda_0 = \max(\text{abs}(\boldsymbol{c})), \quad (78)$$

and the support $\mathcal{I} \doteq \{i : |c_i| = \lambda\}$.

At each iteration k, first assuming $\boldsymbol{w}$ is fixed, the update direction for $\boldsymbol{e}$ is

$$\boldsymbol{d}^{(k)}(\mathcal{I}) = -\lambda \cdot \text{sgn}(\boldsymbol{c}^{(k)}(\mathcal{I})), \quad (79)$$

and the direction for the coefficients that are not in the support $\mathcal{I}$ is manually set to zero. Hence,

$$\boldsymbol{e}^{(k+1)} = \boldsymbol{e}^{(k)} + \gamma \boldsymbol{d}^{(k)}. \quad (80)$$

In (80), a proper step size value γ remains to be determined. In Homotopy methods, γ is set as the minimal step size that leads to either a coefficient of $\boldsymbol{c}$ that is in the support $\mathcal{I}$ crosses zero, and hence should be removed from $\mathcal{I}$; or the magnitude of a coefficient of $\boldsymbol{c}$ outside the support $\mathcal{I}$ reaches or exceed λ, and hence should be added to $\mathcal{I}$.

The first situation is easy to evaluate:

$$\gamma^- = \min_{i \in \mathcal{I}} \{-e_i/d_i\}. \quad (81)$$

However, choosing γ^+ for the second situation presents a problem, as in (77) any update in $\boldsymbol{e}$ does not change the values of $\boldsymbol{c}$ outside the support. In other words, due to the fact that the dictionary for $\boldsymbol{e}$ where its coefficients are sparse is indeed an *identity* matrix, the original strategy in Homotopy could not add any new indices into the support of $\boldsymbol{e}$. An alternative solution that we use in our implementation is to determine γ and $\boldsymbol{w}$ separately using the first constraint (76), and then iterate.

7.3 Solving Face Alignment via Iterative Soft-Thresholding

The composite objective function $F(\boldsymbol{w}, \boldsymbol{e}) = f(\boldsymbol{w}, \boldsymbol{e}) + \lambda g(\boldsymbol{e})$ in (75) can be also solved by *iterative soft-thresholding* (IST) methods. To simplify the notation, denote $\boldsymbol{y} = [\boldsymbol{w}; \boldsymbol{e}] \in \mathbb{R}^{n+d}$. Then the objective function becomes:

$$F(\boldsymbol{y}) = \frac{1}{2}\|\boldsymbol{b} - [B, I]\boldsymbol{y}\|^2 + \lambda\|\boldsymbol{e}\|_1. \tag{82}$$

Then the update rule to minimize $F(\boldsymbol{y})$ is computed by substitutting a linearized approximation of f as shown below:

$$\begin{aligned} rcl\boldsymbol{y}^{(k+1)} &= \arg\min_{\boldsymbol{y}} \{(\boldsymbol{y} - \boldsymbol{y}^{(k)})^T \nabla f(\boldsymbol{y}^{(k)}) + \\ &\quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{y}^{(k)}\|^2 \cdot \nabla^2 f(\boldsymbol{y}^{(k)}) + \lambda g(\boldsymbol{e})\}. \end{aligned} \tag{83}$$

We further introduce an intermediate variable

$$\boldsymbol{u}^{(k)} = \boldsymbol{y}^{(k)} - \frac{1}{\alpha^{(k)}} \nabla f(\boldsymbol{y}^{(k)}), \tag{84}$$

where the hessian $\nabla^2 f(\boldsymbol{y}^{(k)})$ is approximated by the diagonal matrix $\alpha^{(k)} I$. Then based on the IST approach, (83) has a closed-form solution as follows: For $i \leq n$,

$$y_i^{(k+1)} = w_i^{(k+1)} = \arg\min_{y_i} \left\{ \frac{(y_i - u_i^{(k)})^2}{2} \right\} = u_i^{(k)}; \tag{85}$$

and for $i > n$,

$$\begin{aligned} y_i^{(k+1)} &= e_{i-n}^{(k+1)} \\ &= \arg\min_{y_i} \left\{ \tfrac{(y_i - u_i^{(k)})^2}{2} + \tfrac{\lambda |y_i|}{\alpha^{(k)}} \right\} \\ &= \text{soft}\left(u_i^{(k)}, \tfrac{\lambda}{\alpha^{(k)}} \right). \end{aligned} \tag{86}$$

Acknowledgements

The authors would like to thank Dr. John Wright and Dr. Zhouchen Lin at Microsoft Research Asia for his valuable comments. Yang also appreciates the hospitality of the Visual Computing Group at Microsoft Research Asia during his visit there in 2009.

References

[1] M. Asif. Primal dual prusuit: A homotopy based algorithm for the dantzig selector. M. S. Thesis, Georgia Institute of Technology, 2008.

[2] D. Baron, M. Wakin, M. Duarte, S. Sarvotham, and R. Baraniuk. Distributed compressed sensing. *preprint*, 2005.

[3] J. Barzilai and J. Borwein. Two point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

[4] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[6] S. Becker, J. Bobin, and E. Candes. NESTA: a fast and accurate first-order method for sparse recovery. *preprint*, 2009.

[7] P. Belhumeur, J. Hespanda, and D. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[8] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2003.

[9] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[10] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[11] E. Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians*, 2006.

[12] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Math*, 59(8):1207–1223, 2006.

[13] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[14] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4:1168–1200, 2005.

[15] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[16] I. Daubechies, M. Defrise, and C. Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Math*, 57:1413–1457, 2004.

[17] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Journal of Constructive Approximation*, 13:57–98, 1997.

[18] D. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41:613–627, 1995.

[19] D. Donoho. For most large underdetermined systems of linear equations the minimal ℓ^1-norm near solution approximates the sparest solution. *Communications on Pure and Applied Mathematics*, 59(7):907–934, 2006.

[20] D. Donoho, A. Maleki, and A. Montanari. Message-passing algorithms for compressed sensing. *PNAS*, 106(45):18914–18919, 2009.

[21] D. Donoho and Y. Tsaig. Fast solution of ℓ^1-norm minimization problems when the solution may be sparse. preprint, *http://www.stanford.edu/ tsaig/research.html*, 2006.

[22] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[23] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[24] K. Frisch. The logarithmic potential method of convex programming. Technical report, University Institute of Economics (Oslo, Norway), 1955.

[25] R. Gross, I. Mathews, J. Cohn, T. Kanade, and S. Baker. Multi-PIE. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2006.

[26] E. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for ℓ^1-regularized minimization with applications to compressed sensing. Technical Report CAAM Technical Report TR07-07, Rice University, Houston, TX, 2007.

[27] D. Hertog, C. Roos, and T. Terlaky. On the classical logarithmic barrier function method for a class of smooth convex programming problems. *Journal of Optimization Theory and Applications*, 73(1):1–25, 1992.

[28] J. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms*. Springer-Verlag, 1996.

[29] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[30] C. Kelley. *Iterative methods for linear and nonlinear equations*. SIAM, Philadelphia, 1995.

[31] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[32] M. Kojima, N. Megiddo, and S. Mizuno. Theoretical convergence of large-step primal-dual interior point algorithms for linear programming. *Mathematical Programming*, 59:1–21, 1993.

[33] A. Maleki and D. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *preprint*, 2009.

[34] D. Malioutov, M. Cetin, and A. Willsky. Homotopy continuation for sparse signal representation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.

[35] N. Megiddo. Pathways to the optimal set in linear programming. In *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 131–158, 1989.

[36] R. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part I: Linear programming. *Mathematical Programming*, 44:27–41, 1989.

[37] D. Needell and J. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harmonic Anal.*, 26:301–321, 2008.

[38] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[39] Y. Nesterov. Gradient methods for minimizing composite objective function. *ECORE Discussion Paper*, 2007.

[40] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, 2rd edition, 2006.

[41] M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–404, 2000.

[42] M. Plumbley. Recovery of sparse representations by polytope faces pursuit. In *Proceedings of International Conference on Independent Component Analysis and Blind Source Separation*, pages 206–213, 2006.

[43] T. Serafini, G. Zanghirati, and L. Zanni. Gradient projection methods for large quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20(2–3):353–378, 2004.

[44] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996.

[45] J. Tropp and S. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98:948–958, 2010.

[46] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, and Yi Ma. Toward a practical face recognition: Robust pose and illumination via sparse representation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2009.

[47] J. Wright and Y. Ma. Dense error correction via ℓ^1-minimization. *IEEE Transactions on Information Theory*, 56(7):3540–3560, 2010.

[48] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[49] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210 – 227, 2009.

[50] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.

[51] A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Fast l1-minimization algorithms and an application in robust face recognition: a review. In *Proceedings of the International Conference on Image Processing*, 2010.

[52] A. Yang, M. Gastpar, R. Bajcsy, and S. Sastry. Distributed sensor perception via sparse representation. *Proceedings of the IEEE*, 98(6):1077–1088, 2010.

[53] J. Yang and Y. Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *(preprint) arXiv:0912.1185*, 2009.

[54] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.